APPENDIX B

Setting Up the UNIX™ System (DEC)

Setting up the UNIX™ System (DEC)

GENERAL

This section describes the load, update, and configuration procedures involved in installing a UNIX[1] operating system on the following DEC processors:

- Digital Equipment Corp. VAX 11/780, 11/750
- Digital Equipment Corp. PDP 11/70

A. Prerequisites

Before attempting to generate a UNIX operating system, the system administrator should understand that a considerable knowledge of the related documentation is required and assumed. In particular, the administrator should be very familiar with the following documents:

- *UNIX System User's Guide*
- *UNIX System User's Manual*
- *UNIX System Administrator's Manual*
- *UNIX System Operator's Guide*
- All sections in the *UNIX System Administrator's Guide.*

Throughout this section, each reference of the form **name**(1M), **name**(7), or **name**(8) refers to entries in the *UNIX System Administrator's Manual.* All other references to entries of the form **name**(N), where "N" is a number(1 through 6) possibly followed by a letter, refer to entry **name** in section N of the *UNIX System User's Manual* .

The system administrator must have a basic understanding of the operation of the hardware. This includes the operation of the console and the tape and disk drives, which are assumed to have standard UNIX system addresses and interrupt vectors. It is also assumed that the hardware works and has been completely installed. All appropriate DEC diagnostics should have been run to test the configuration, and a detailed description of the hardware including device addresses, interrupt vectors, and bus levels is needed. This information is necessary to generate a UNIX system.

B. Procedure

The UNIX system is distributed on two magnetic tapes, recorded in 9-track format at either 800 bpi or 1600 bpi. Distribution tapes will be marked either "PDP-11" or "VAX", 800 bpi or 1600 bpi. Make sure you have the correct tape for your machine. UNIX System V is to be installed using only the *initial load procedures* described in this document.

Initial Load

The initial load program (on Tape 1) will copy a file system from tape (VAX-11/780: either a TE16 or a TU78; VAX-11/750: either a TS11 or a TU77; PDP-11: either a TU10 or a TU16) to disk (VAX: either a RP06, a RP07, a RM05, or a RM80; PDP-11: either an RP03, an RP06, or an RM05). In this document, RP04/5 drives are considered to be equivalent to RP06 drives; any differences will be noted explicitly. Once the *root* file system has been successfully loaded to disk, the UNIX system may be booted and the available utility programs used to complete the installation.

Update

The update procedures are to be used *only* on releases that are designated as *update* releases. The **cpio**(1) program is used to perform all updates. The **cpio** program will not update any file if its replacement has a modification time that is less than (i.e., earlier than) the modification time of the

1. UNIX is a trademark of Bell Telephone Laboratories, Incorporated.

original file. Certain administrative files (e.g., */etc/passwd*) are sent with a modification time of January 1, 1970 to ensure that they do not replace their counterparts during updates. Any file not copied will cause **cpio** to print a message to that effect. These messages should always be investigated to ensure that any files not copied were of that type. However, note that, depending on respective modification times, a locally-modified file may get updated, thus destroying the local modifications.

One of the most common problems that can arise during an installation is running out of disk space when performing an update. Should this occur, the original contents of the file system should be restored from a backup copy and the contents of the update tape should be read into a spare file system using the **cpio** program. Unwanted material can then be removed and the original file system can be updated from this new file system using the **−pdm** options of **cpio.**

LOAD PROCEDURES

A. Distribution Tape Format

Tape 1 contains eight files: a loader, a physical copy of the *root* file system, the **cpio** program, a **cpio** structured copy of the *root* file system, and four files (**cpio** format) that represent selectable items. *Root* refers to the directory "/", which is the root of all the directory trees. The format of this tape is as follows:

file 1:	Tape boot loader — 512 bytes;
	Tape boot loader — 512 bytes;
	Initial load program — several 512-byte records;

file 2:	*root* file system (physical) — 5120-byte records (blocking factor 10);
file 3:	**cpio** program (latest version) — several 512-byte records (to be used *only* for updating an earlier UNIX system release);
file 4:	the *root* file system (structured in **cpio** format) — several 5120-byte records (to be used *only* for updating an earlier UNIX system release);
file 5:	on-line manual pages (same format as file 4);
file 6:	on-line documents (same format as file 4);
file 7:	RJE software (same format as file 4);
file 8:	graphics software (same format as file 4).

The *root* (/) file system contains the following directories:

bck:	Directory used to mount a backup file system for file restoring.
bin:	Public commands; described in Section 1 of the *UNIX System User's Manual*
dev:	Special files, all the devices on the system.
etc:	Administrative programs and tables.
lib:	Public libraries, parts of the assembler, C compiler.
mnt:	Directory used to mount a file system.
lost+found:	Directory used by **fsck**(1M) for disconnected files.
stand:	Stand-alone programs.
tmp:	Directory used for temporary files; should be cleaned at reboot.
usr:	Directory used to mount the */usr* file system.

B. Initial Load of Root

Mount Tape 1 on drive 0 and position it at the load point.

PDP-11

Boot the tape by reading either record 0 or 1 into memory starting at address 0 and start execution at address 0. This may be accomplished by using a standard DEC ROM bootstrap loader, a special ROM, or some manual procedure; see **romboot**(8), **tapeboot**(8), and **70boot**(8).

VAX

See "Installation Boot Procedures" under either **11/780 ops**(8) or **11/750 ops**(8). These entries describe initial tape booting and modification of the console floppy disk to simplify UNIX system administration.

C. Common to PDP-11 and VAX

The tape boot loader will type "UNIX tape boot loader" on the console terminal and read in and execute the initial load program. The program will then type detailed instructions about the operation of the program on the console terminal. The program will ask what type of disk drive you have and which drive you plan to use for the copy. The disk controller used must be at the standard DEC address indicated by the program; however, other disk controllers on your system may be at nonstandard addresses. A formatted, ECC flag-free pack must be mounted on the drive you have indicated. If necessary, use the appropriate DEC diagnostic program to format the pack. For the VAX use **format**(1M). Note that the pack will be written on. Next, the program will ask what type of tape drive you have and which drive contains the tape. Normally, this will be drive 0, but the program will work with other drives. Note that the tape is currently positioned correctly after the end-of-file between the initial load program and the *root* file system. When everything is ready, the program will copy the file system from the tape to disk and give instructions for booting the UNIX system. After the copy is complete and you have booted the basic version of the UNIX operating system, check [using **fsck**(1M)] the *root* file system and browse through it.

PDP-11/70 Only

The file */stand/mmtest* is a stand-alone memory mapping diagnostic program for the PDP-11/70. If you are not *absolutely* sure that DEC FCO (field change order) M8140-R002 has been applied to your PDP-11/70 CPU, *stand/mmtest* should be booted and allowed to run at least 20 minutes. To boot this program, go through the disk boot procedure, but specify:

```
    0=stand/mmtest
```

The UNIX system comes with optional power-fail recovery. This feature *requires* that the *power-up* and *power-down* interrupt vectors be 024.

D. Update of Root

It is very important that the system be running in *single-user* mode during the update phase. To update an already existing *root* file system, files three and four on Tape 1 will be used. It is necessary to first make a copy of the *root* file system using **volcopy**(1M) and then update this copy. The copy should be made on a separate disk pack using the same section number as the *root* file system (always section 0). Also, after the update is completed, check if any of the local administrative files in the directory */etc* need modification. Most of these are mentioned in the part "ADMINISTRATIVE FILES".

Mount Tape 1 on drive 0 and position it at the load point. It is assumed that disk drive 1 is available for making the copy and that the *root* file system is on */dev/rp0*. The following procedure will first make a copy of the *root* file system, and then update this copy. Note that */dev/nmt4* refers to tape drive 0 but has the side effect of spacing forward to the next end-of-file (no rewind option). The **−B** option of **cpio** specifies that input is in 5120-byte records:

```
volcopy root /dev/rrp0 pkname1 /dev/rrp10 pkname2
mount /dev/rp10 /bck
#  The 2 echoes will move the tape to file 3
echo </dev/mt4
echo </dev/mt4
cp /dev/mt4 /bck/bin/cpio
chmod 755 /bck/bin/cpio
chown bin /bck/bin/cpio
cd /bck
/bck/bin/cpio −idmB </dev/rmt0
cd /
umount /dev/rp10
```

Pkname1 and *pkname2* are the volume names of the source and destination disk packs, respectively. If the new copy is satisfactory, shut down and halt the system, change disk packs, and reboot the system using the new *root*.

E. Tape 2 (/usr) Format

Tape 2 contains the */usr* file system in **cpio** format (5120-byte records). The */usr* file system contains commands and files that must be available (mounted) when the system is in *multiuser* mode. The tape contains the following directories:

adm:	Miscellaneous administrative command and data files including the process accounting file *pacct*.
bin:	Public commands; an overflow for */bin*.
games:	Various demonstration and instructional programs.
include:	Public C language *#include* files.
lib:	Archive libraries including the text processing macros; also contains data files for various programs such as **spell**(1) and **cron**(1M).
mail:	Mail directory.
lost+found:	Directory used by **fsck**(1M) for disconnected files.
news:	Place for all the various system news; see **news**(1).
pub:	Handy public information, e.g., table of ASCII characters.
spool:	Spool directory for daemons.
src:	Source for commands, libraries, the system, etc.
tmp:	Directory for temporary files; should be cleaned at reboot.

F. Initial Load of /usr

Mount Tape 2 on drive 0 and position it at the load point. Mount a file system (device) as */usr*. The ultimate size and location of this file system on a device is an administrative decision; initially, the following procedure will suffice:

```
mkfs /dev/rrp1 65000 gap blocks
#  See mkfs(1M) for appropriate parameters.
#  Note that for RP03 disks, this will use part of section 2.
labelit /dev/rrp1 usr pkname
mount /dev/rp1 /usr
chmod 775 /usr
cd /usr
cpio −idmB </dev/rmt0
```

Pkname is the volume name of the pack (e.g., "p0001").

Because */usr* must be mounted when the system is in *multiuser* mode, the file */etc/rc* must be changed to include the command lines to mount and unmount the file systems in *single user* and *multiuser* mode. These lines must be inserted at the appropriate places in */etc/rc* as indicated by comments in the prototype file. Next, the file */etc/checklist* should be changed to include the file system device (e.g., */dev/rrp1*); see **fsck**(1M), **labelit**(1M), **mkfs**(1M), **mount**(1M), and **checklist**(4).

G. Update of /usr

It is advisable that the system be running in *single*user mode during the update phase. It is also wise to first make a copy of your */usr* file system for backup purposes. Next, mount Tape 2 on drive 0 and position it at the load point. The */usr* file system *must* also be mounted. The following procedure will perform the update:

```
cd /usr
cpio −idmB </dev/rmt0
```

H. Initial Load or Update of Selectable Items

The initial load and update procedures are essentially the same; the only exception being the creation of the selectable item directory on the initial load.

Mount Tape 1 on drive 0 and position it at the load point. Make sure that the */usr* file system is mounted. The following procedure will read in the source for each of the selectable subsystems. If a particular subsystem is not desired, simply skip that file on the tape by executing the following command:

```
echo </dev/rmt4
```

The tape can be rewound after any subsystem by specifying */dev/rmt0* instead of */dev/rmt4*.

```
echo </dev/rmt4;  echo </dev/rmt4
echo </dev/rmt4;  echo </dev/rmt4

cd /usr
mkdir man; chown bin man; chgrp bin man; chmod 775 man
cd man
cpio −idmB </dev/rmt4

cd /usr
mkdir docs; chown bin docs; chgrp bin docs; chmod 775 docs
cd docs
cpio −idmB </dev/rmt4

cd /usr/src/cmd
mkdir rje; chown bin rje; chgrp bin rje; chmod 775 rje
cd rje
cpio −idmB </dev/rmt4

cd /usr/src/cmd
mkdir graf; chown bin graf; chgrp bin graf; chmod 775 graf
cd graf
cpio −idmB </dev/rmt0
```

After installing the source for the *rje* and *graphics* subsystems, the software must be *built* and *installed*. (Execute the commands for the subsystems that you have elected to take.)

To build and install *rje*, change your working directory to */usr/src* and execute one of the following **:mkcmd** lines:

```
ARGS="rje1"     ./:mkcmd rje     # makes a single IBM system
ARGS="rje2"     ./:mkcmd rje     # makes rje1, and rje2
ARGS="rje3"     ./:mkcmd rje     # makes rje1, rje2, and rje3
ARGS="rje4"     ./:mkcmd rje     # makes rje1, rje2, rje3, and rje4
```

See **rje**(8) and the "UNIX System Remote Job Entry " section for additional information.

To build and install the graphics package:

```
cd /usr/src
./:mkcmd graf
```

CONFIGURATION PLANNING

A. UNIX System Configuration

The basic UNIX operating system supplied supports only the console, a disk controller (disk drive 0), and a tape controller (tape drive 0). Each system administrator must describe the actual configuration of their own system.

All of the UNIX operating system source code and object libraries are in */usr/src/uts*. All of the configuration information is kept in the directory */usr/src/uts/*/cf*; the "*" represents either *pdp11* or *vax*. There are only two files that must be changed to reflect your system configuration - *low.s* (*univec.c* on the VAX) and *conf.c*. The program **config**(1M) should be used to make these changes.

Config requires a "system description file" and produces the two needed files. Table 4.A lists the values and sizes of the basic parameters for the different CPUs. For more details of syntax and structure, see **config**(1M) and the associated **master**(4).

The first part of the system description file lists all of the hardware devices on the system. Next, various system information is listed. A brief explanation of this information follows:

TABLE 4.A.

Item	PDP-11/70		VAX-11/750,/780	
	Range	Size	Range	Size
nswap	3000	—	9000	—
buffers	25-60	30†	80-400	56‡
sabufs	10-15	542	—	—
hashbuf	32-128	6	32-128	12
physbuf	3-5	30	3-7	56
inodes	100-250	24	100-300	84
iblocks	80-200	52	—	—
files	100-250	8	100-300	12
mounts	8-16	8	8-20	20
coremap	50-100	4	—	—
swapmap	50-100	4	50-100	4
calls	30-60	6	30-60	12
procs	50-200	32	50-200	64
texts	25-50	12	25-50	16
clists	100-300	28	100-250	72
maxproc	25	—	25	—

† Plus 512 bytes/buffer outside system space.

‡ Plus 1024 bytes/buffer allocated at start up.

- *root*—— Specifies the device where the *root* file system is to be found. The device must be a block device with read/write capability because this device will be mounted read/write as "/". Thus, a tape can not be mounted as the *root* but can be mounted as some read-only file system. Normally, *root* is disk drive 0, section 0.

- *pipe*—— Specifies where pipes are to be allocated (must be a *mounted* file system — the root file system is normally used).

- *dump*—— Specifies the device to be used to dump memory after a system crash. Currently, only the TU10, TU16/TE16, TU78, and TS11 tape drives are supported for this purpose.

- *swap*—— Specifies the device and blocks that will be used for *swapping*. *Swplo* is the first block number used and *nswap* indicates how many blocks, starting at *swplo*, to use. Care must be taken that the swap area specified does not overlap any file system. For example, if section 0 is 8000 blocks long, the *root* file system could occupy the first 6000 blocks, and *swap* the remaining 2000 by specifying:

```
root rp06 0
swap rp06 0 6000 2000
```

- *buffers*——Specifies how many "system buffers" to allocate. Real-time response improves as more buffers are allocated. UNIX system buffers form a "data cache". Improvement in the hit rate of this cache tends to fall as the number of buffers is increased.

- *sabufs*——PDP-11 only: specifies how many "system addressable" buffers to allocate. One buffer is needed for every mounted file system. Certain I/O drivers need such buffers.

- *hashbuf*——Specifies how many hash buckets to allocate. These are used to search for a buffer given a device number and block number. This number must be a power of two. The default value is 64.

- *physbuf*——Specifies how many physical I/O buffer headers to allocate. One is needed for each physical read or write active. The default value is 4.

- *inodes*——Specifies how many "inode table" entries to allocate. Each entry represents a unique open inode. When the table overflows, the warning message "Inode table overflow" will be printed on the console. The table size should be increased if this happens regularly. The number of entries used depends on the number of active processes, texts, and mounts.
- *iblocks*——PDP-11 only: specifies how many "inode block address cache" entries to allocate. An entry is needed for each regular, directory, or fifo file that is open. Since special files do not need an entry, data space can be conserved by specifying a much smaller number of iblocks than inodes. The default value is four less than the number of inodes.
- *files*——Specifies how many "open-file table" entries to allocate. Each entry represents an open file. When the table overflows, the warning message "no file" will be printed on the console. The table size should be increased if this happens regularly.
- *mounts*——Specifies how many "mount table" entries to allocate. Each entry represents a mounted file system. The *root* (/) will always be the first entry. When full, the **mount**(2) system call will return the error EBUSY.
- *coremap*——PDP-11 only: specifies how many entries to allocate to the "list of free memory". Each entry represents a contiguous group of 64-byte blocks of free memory. When the list overflows, due to excessive fragmentation, the system will print a warning message on the console. This condition results in the loss of memory being freed. The system should be remade with a larger table size and rebooted. The number of entries used depends on the number of processes active, their sizes, and the amount of memory available.
- *swapmap*——Specifies how many entries to allocate to the "list of free swap blocks". Exactly like the *coremap*, except it represents free blocks in the swap area, in 512-byte units.
- *calls*——Specifies how many "call-out table" entries to allocate. Each entry represents a function to be invoked at a later time by the clock handler. The time unit is 1/60 of a second. The call-out table is used by the terminal handlers to provide terminal delays and by various other I/O handlers. When the table overflows, the system will crash and print the panic message "Timeout table overflow" on the console. This value must be greater than two.
- *procs*——Specifies how many "process table" entries to allocate. Each entry represents an active process. The scheduler is always the first entry and **init**(1M) is always the second entry. The number of entries depends on the number of terminal lines available and the number of processes spawned by each user. The average number of processes per user is in the range of 2 through 5. When full, the **fork**(2) system call will return the error EAGAIN.
- *texts*——Specifies how many "text table" entries to allocate. Each entry represents an active read-only text segment. Such programs are created by using the **−i** or **−n** option of the loader **ld**(1). The **−n** option is implicit on the VAX. When the table overflows, the warning message "out of text" is printed on the console.
- *clists*——Specifies how many "character list buffers" to allocate. On the PDP-11, each buffer contains up to 24 bytes; on the VAX, each buffer contains up to 64 bytes. The buffers are dynamically linked together to form input and output queues for the terminal lines and various other slow-speed devices. The average number of buffers needed per terminal line is in the range of 5 through 10. When full, input characters from terminals will be lost and not echoed.
- *maxproc*——Specifies how many concurrent processes a non-superuser is allowed to run.
- *power*——Specifies whether to attempt restart after a power failure. A value of 0 (default) indicates no restart; a value of 1 attempts power-fail restart. On restart, device drivers are called and process 1 (i.e., **init**) is sent a hangup signal; see **init**(1M).

- *sema*——Specifies whether to include semaphore code. A value of 0 (default) indicates no semaphores; a value of 1 includes semaphores.
- *shmem*——Specifies whether to include shared memory code. A value of 0 (default) indicates no shared memory; a value of 1 includes shared memory.
- *mesg*——Specifies whether to include message code. A value of 0 (default) indicates no messages; a value of 1 includes messages.
- *shmmax*——Specifies the maximum size of a shared memory segment.
- *shmmin*——Specifies the minimum size of a shared memory segment.
- *shmmni*——Specifies the maximum number of shared memory segments in the system.
- *shmseg*——Specifies the maximum number of shared memory segments a user may have attached.
- *shmall*——Specifies the maximum amount of shared memory that may be allocated system wide. The default value is 512 clicks, 250k bytes.
- *shmbrk*——Specifies the number of clicks between the end of the data segment, and the beginning of the first shared memory segment if the default starting address is used allowing the user to continue to use **sbrk**(2) or **brk**(2). The default value is 16 clicks, 8k bytes.
- *msgmax*——Specifies the maximum message size.
- *msgmnb*——Specifies the maximum number of bytes on any one queue.
- *msgtql*——Specifies the number of system message headers, i.e., maximum number of outstanding messages.
- *msgssz*——Specifies the message segment size. Messages consist of a set of contiguous message segments large enough to fit the text. The segments are used to help eliminate fragmentation and speed message buffer allocation. A message may span several segments.
- *msgseg*——Specifies the number of message segments in the system.
- *msgmap*——Specifies the message segment map size.
- *msgmni*——Specifies the maximum number of message queues system wide. The default is 10.
- *semmap*——Specifies the number of entries in the semaphore map. The map is used by the system to allocate and free semaphore sets. This parameter should be changed to reflect changes in *semmns*.
- *semmni*——Specifies the number of semaphore identifiers, i.e., number of semaphore sets.
- *semmns*——Specifies the number of semaphores in the system.
- *semmnu*——Specifies the number of undo structures in the system.
- *semume*——Specifies the maximum number of undo entries per structure.
- *semmsl*——Specifies the maximum number of semaphores per semaphore identifier.
- *semopm*——Specifies the maximum number of semaphore operations per **semop**(2) call.
- *maus*——PDP-11 only: specifies whether to include maus (shared memory) code. A value of 0 (default) indicates no shared memory; a value of 1 includes the shared memory code.

B. UNIX System Generation

Before generating the ***first*** UNIX system, the system administrator should modify the file called *Makefile* in the */usr/src/uts/*/cf* directory. This file contains five symbols that are used for system identification; it is used to initialize the internal *utsname* structure [see **uname**(1) and **uname**(2)].

The five symbols (which are eight characters maximum) are as follows:

SYS	System name (e.g., *pwba*);
NODE	The name by which the system is known on the **uucp**(1C) network (e.g., *pwba*);
REL	The operating system release (e.g., *sysV*);
VER	The current version of the system; this is usually four characters indicating when the system was made (e.g., *0620* for June 20);
MACH	The machine hardware name (e.g., *vax-780*).

Generally, only the first two symbols need local modification; the REL symbol is a constant for the duration of this release, while the VER symbol will be defined when you **make**(1) the system. The name of the executable file produced by the generation procedure will be the concatenation of the SYS and VER symbols (e.g., *pwba0620*).

To generate a new UNIX operating system, follow the procedure given:

```
cd /usr/src/uts/*/cf
ed dfile
■
   [information as described above]
.
w
q
config dfile
make VER=ver
```

The PDP-11 system has a relatively small address space. If the table sizes or the number of device types are too large, the program **sysfix** will print various error messages and the above procedure will only create an *a.out* file. In particular, the maximum available data space is 49,152 bytes. The actual data space requested can be found by using **size**(1) on *a.out* and adding the *data* and *bss* segment sizes. One then reduces the specified values for the various system entries until it all fits. The amount of space in the *bss* segment used for each entry is indicated in the part "UNIX System Configuration".

On the VAX, the combined data space should not exceed 200,000 bytes.

The PDP-11 system is distributed with a special *overlay loader* that allows larger systems (text size greater than 64K) to be configured. This system will be made up of one main segment and seven overlay text segments that use *supervisor* registers to switch the overlay text segments. The main segment may be as large as 56K, and the overlay text segments may not be greater than 8K each. To invoke this new text scheme, edit */usr/src/uts/*/Makefile*. To change the TYPE symbol from *id* to *ov* or when generating a new system, use the following procedure:

```
cd /usr/src/uts/*/cf
make VER=ver TYPE=ov
```

When you are satisfied with the new system, test it by the following procedure:

```
cp /usr/src/uts/*/sysver  /   # sysver as above
cd /
umount /dev/rp1
rm /unix
ln /sysver /unix
sync
```

Halt the processor and reboot the system. Note that this procedure results in two names for the operating system object the generic */unix* and the actual name, say */pwba0620*. An old system may be booted by referring to the actual name, but remember that many programs [such as **ps**(1)] use the generic name */unix* to obtain the *name-list* of the system.

If the new system does not work, verify that the correct device addresses and interrupt vectors have been specified. On the PDP-11, if the *wrong* interrupt vector and the *correct* device address have been specified for a device, the operating system will print the error message "stray interrupt at XXX" when the device is accessed, where XXX is the correct interrupt vector. On the VAX, the message is "stray UBA interrupt at XXX". On the PDP-11, if the *wrong* device address is specified, the system will crash with a panic trap of type 0 (indicating a time-out) when the device is accessed. On the VAX, the system will not crash directly but will print a "UBA" warning message indicating the failing address.

For the VAX, a stand-alone test may be executed to compare the configuration in */unix* with the actual hardware. After the stand-alone shell prompts "$$", type */stand/vcf*. The **vcf**(1M) command reports on the MASSBUS adapter configuration and attempts to determine the UNIBUS system device configuration. Any errors should be corrected before continuing.

C. Special Files

A special file must be made for every device on your system. Normally, all special files are located in the directory */dev*. Initially, this directory will contain:

console	console terminal
error	see **err**(7)
mem, kmem, null	see **mem**(7)
tty	see **tty**(7)
rp[0-7], rrp[0-7]	disk drive 0, sections 0-7
rl[0-1], rrl[0-1]	disk drives 0 and 1
rk[0-1], rrk[0-1]	disk drives 0 and 1
mt0, rmt0	tape drive 0
mt4, rmt4	tape drive 0 (no rewind).

These special files are of two types - block and character. This is indicated by the character *b* or *c* in the listing produced by **ls**(1) with the **−l** option.

In addition, each special file has a major device number and a minor device number. The major device number refers to the device type and is used as an index into either the *bdevsw* or *cdevsw* table in the configuration file *conf.c* The minor device number refers to a particular unit of the device type and is used only by the driver for that type. The **config** program with the −t option will list major device numbers.

The program **mknod**(1M) creates special files. For example, the following would create *part* of the initially-supplied */dev* directory (on the PDP-11):

```
cd /dev
mknod console c 0 0
mknod error c 20 0
mknod mem c 2 0; mknod kmem c 2 1; mknod null c 2 2
mknod tty c 13 0
mknod rp0 b 0 0; mknod rrp0 c 7 0
mknod mt0 b 1 0; mknod rmt0 c 6 0
mknod mt4 b 1 4; mknod rmt4 c 6 4
```

After the special files have been made, their access modes should be changed to appropriate values by **chmod**(1). For example:

```
cd /dev
chmod 622 console
chmod 444 error
chmod 440 mem kmem
chmod 666 null
chmod 666 tty
chmod 400 rp0 rrp0
chmod 666 mt0 rmt0
chmod 666 mt4 rmt4
```

Note that file names have no meaning to the *operating system* itself; only the major and minor device numbers are important. However, many *programs* expect that a particular file is a certain device. Thus, by convention, special files are named as follows:

block device	*conf.c*	*/dev*
RP03 disk	rp	rp*
RP04/5/6 disk	hp	rp*
RS03/4 fixed head disk	hs	rs*
RK05 disk	rk	rk*
RL01/2 disk	rl	rl*
RM05 disk	hm	rp*
RM80 disk	gd	rp*
RP07 disk	gd	rp*
general disk	gd	rp*
TU10 tape	tm	mt*
TE/TU16 tape	ht	mt*
TS11 tape	ts	mt*
TU78 tape	ts	mt*
general tape	gt	mt*

character device	*conf.c*	*/dev*
DL11 async. line	kl	tty*, console
DH11 async. line mux	dh	tty*
DMC11 sync. unit	dmc	dmc*
DZ11 async. line mux	dz	tty*
DN11 auto call unit	dn	dn*
DU11 sync. line	du	du*
KMC11-B micro	kmc	kmc*
DM11-BA modem control	dmk	dmk*
DZ11/KMC11-B assist	dzb	tty*
LP11 line printer	lp	lp*
RP03 disk	rp	rrp*
RP04/5/6 disk	hp	rrp*
RM05 disk	hm	rrp*
RM80 disk	gt	rrp*
RP07 disk	gt	rrp*
general disk	gd	rrp*
RS03/4 fixed head disk	hs	rrs*
TU10 tape	tm	rmt*
TE/TU16 tape	ht	rmt*
TS11 tape	ts	rmt*
TU78 tape	gt	rmt*
general tape	gt	rmt*
error	err	error
memory	mm	mem, kmem, null
terminal	sy	tty

For those devices with a */dev* name ending in "*", this character is replaced by a string of digits representing the *minor* device number. For example:

```
mknod /dev/mt1 b 1 1
mknod /dev/rp24 b 0 024 # leading zero means octal
mknod /dev/tty03 c 1 3
```

Note that for disks, an octal number scheme is maintained because each drive is logically partitioned into eight sections. Thus, */dev/rp24* refers to section 4 of physical drive 2. There is also a special file, */dev/swap*, that is used by the program **ps**(1). This file must reflect what *block* device is used for swapping and must be readable. For example:

```
rm /dev/swap
mknod /dev/swap b 0 0
chmod 440 /dev/swap
chown sys /dev/swap; chgrp sys /dev/swap
```

The minor device numbers for tapes are also encoded; the minor device number consists of the four bits shown in Fig. 4.1.

DENSITY SELECT	0=REWIND 1=NO REWIND	DRIVE-SELECT BIT 1	DRIVE-SELECT BIT 0

Minor Device Number, Tape

FIG. 4.1 -

For the TU10 and TE16, the density select bit is 0 for 800 bpi and 1 for 1600 bpi. For the TU78, the density select bit is 0 for 1600 bpi and 1 for 6250 bpi. For the TS11, the density select bit is ignored; the density is always 1600 bpi. Therefore, the special file for tape drive 1, for operation at 1600-bpi, with no rewind on close would have a minor device number of 13 (binary 1101). The 2-bit drive number field permits a maximum of four drives per system.

The minor numbers of the automatic calling unit (ACU) interface, DN11, are encoded in eight bits as shown in Fig. 4.2.

SHARED-ACU-SELECT FOUR BITS	SYSTEM-UNIT-SELECT DN11-AA TWO BITS	LINE-UNIT-SELECT DN11-DA TWO BITS

Minor Device Number, DN11

FIG. 4.2 -

The *shared-ACU-select* field in normally zero, which indicates the standard 801 ACU hardware (one ACU, one phone line). For the new shared ACU hardware (one ACU, several phone lines), the *shared-ACU-select* field indicates which line is to be used.

D. File Systems

Each physical pack is partitioned into eight logical sections. This partitioning is defined in the operating system by a table with eight entries. Each table entry is two words long. The first specifies how many blocks are in the section; the second specifies the starting cylinder; see **hp**(7) (RP04/5/6), **hm**(7) (RM05), **rp07**(7) (RP07), **rm80**(7) (RM80), and **rp**(7) (RP03) for default cylinder and block assignments. These values are described to the system in the header file */usr/include/sys/io.h*.

A file system starts at block 0 of a section of the disk and may be as large as the size of that section; if it is smaller than the size of a section, the remainder of that section is unused. Note that the sections themselves may overlap physical areas of the pack, but the file systems must *never* overlap.

The program **mkfs**(1M) (for 1K byte/block file systems) or **omkfs** (for 512 byte/block file systems) is used to initialize a section of the disk to be a file system. The length of each section of the disk is specified in 512 byte sectors. When **mkfs** is used, it produces half the number of 1K byte file system blocks. The 1K byte blocks provide better throughput for the particular file system. A 512 byte/block file system can be made using **omkfs** in place of **mkfs**. The number of physical disk sectors (512 bytes each) used to make the entire file system would be the same for either command. In future releases the functions of **mkfs** and **omkfs** will be merged into one command.

Next, the program **labelit**(1M) is used to label the file system with its name and the name of the pack. Finally, the file system may be checked for consistency by using **fsck**(1M). The file system may then be mounted using **mount**(1M).

E. DZ11 Software with KMC11-B Assist

KMC11-B microprocessors may be used to control DZ11 asynchronous multiplexers, thus off-loading terminal-oriented functions from the CPU. The KMC11-B does DMA input and output of data, character translations, tab expansions, etc. Each KMC11 can control up to four DZ11 multiplexers for a total of 32 asynchronous lines. Each system can support up to four KMC11 microprocessors; however, only 64 DZ11 lines can be controlled by KMC11 microprocessors.

Installation

1. Generate a system by including each DZ11 to be controlled by a KMC11 in the system description file. For example:

   ```
   dzb     X     Y     Z
   ```

   where *X* is the vector address, *Y* is the UNIBUS address, and *Z* is the bus request level. Also include the KMC11 microprocessors in the configuration file:

```
kmc11     X     Y     Z
```

2. Update the files */etc/brc* and */etc/powerfail* to execute **dzbload** before entering one of the numbered *init* states and after a power failure, respectively. Each KMC used to control a DZ11 must be loaded with microcode. For each KMC11 used to control a DZ11, include

   ```
   /etc/dzbload /dev/kmc?
   ```

   where **?** is the minor device number of that KMC11.

3. Special files must be created for each KMC and DZ11 line:

   ```
   /etc/mknod    /dev/kmc?    c    X    ?
   /etc/mknod    /dev/tty??   c    Y    Z
   ```

   where *X* is the major device number of the KMC11 and **?** is the minor device number of the KMC11 controlling the DZ11 multiplexers, i.e., the KMC11 loaded with microcode in Step 2. Y is the major device number of the *dzb* device as is supplied by **config**(1M). Z is the minor device number for a particular line on a DZ11. This number is composed of three fields. The low-order three bits are the line number relative to a DZ11. The next three bits contain the unit number of the DZ11 controlling these lines. Note that this number is the absolute DZ11 number, not the number relative to the KMC. The high-order two bits are the minor device number of the KMC controlling this DZ11. For example:

   ```
   mknod /dev/tty?? c Y 0241
   ```

   specifies the second line (0 through 7) on the fifth DZ11 to be controlled by the KMC11 with minor device number 2. The DZ11 number is specified by the order of appearance in the configuration file. The first four DZ11 multiplexers must be associated with one KMC11 and the next four must be associated with another KMC11. The order in which the KMC11 microprocessors are specified is not significant.

F. Virtual Protocol Machine (VPM) Devices

KMC11-B microprocessors may be used to control DMC11-DA, -FA, or -MD single-line synchronous communications interfaces or DMS11-DA 8-line synchronous communications multiplexers (one per KMC). The combination of a KMC11 and a DMS11-DA is known as a KMS11. The KMC11 executes the link-level communications protocol and performs DMA data transfers between main memory and the communications device. A link-level protocol description in a high-level language is compiled on the host computer [see **vpmc**(1M)] and then down-loaded into the KMC11 by the **vpmstart**(1M) command.

The common synchronous interface (CSI) contains a set of utility routines for communicating with the VPM interpreter in the KMC11. The VPM protocol driver provides a transparent user interface to the VPM interpreter. The CSI routines are used not only by the VPM protocol driver but also by several other protocol drivers.

Installation

1. Generate a system. The system description file must contain a line of the form:

   ```
   vpm     0     0     0     n
   ```

   where *n* is the number of VPM protocol-driver minor devices desired, and a line of the form:

   ```
   csi     0     0     0     m
   ```

   where *m* is the number of CSI minor devices desired. One CSI minor device is required for each VPM protocol-driver minor device. If other drivers that use CSI are configured, additional CSI minor devices must be configured to support these drivers. If the VPM trace capability [see **vpm**(7)] is to be used, the system description file must also contain a line of the form:

```
        trace       0       0       0       k
```

where *k* is the number of trace minor devices desired (normally one). *Vpmbsz*, a configurable parameter which gives the size of the buffer pool available to *vpm*, may be adjusted to be different from the default value found in */etc/master*. Refer to **master**(4) for more information.

2. Compile the protocol script using **vpmc**(1M):

   ```
   vpmc [−m] [−i hdlc[/kms]] −o script.o script.r
   ```

   where *script.r* is the source file for the protocol description. The **−m** option is used only if the protocol source requires expansion with the **m4**(1) preprocessor; otherwise, the C preprocessor is used. The **−i hdlc** option is used to compile a script that uses the bit-oriented (HDLC) version of the VPM interpreter; otherwise, the character-oriented (BISYNC) version of the interpreter will be used. If the script is to be loaded into a KMS11 8-line multiplexer rather than into a KMC11 controlling a single-line interface, the **−i hdlc/kms** option is needed. Only bit-oriented protocols are currently supported by VPM on the KMS11.

3. Install the compiled protocol description in a convenient directory such as */lib* and modify the */etc/rc* file to load the compiled protocol description into the selected KMC11s each time the system is rebooted (first transition to multiuser) using **vpmstart**(1M). If power-fail recovery is desired, the */etc/powerfail* file should also be modified to reload the compiled protocol description into the selected KMC after a power fail occurs. The **−r** option to **vpmstart** is required when reloading the KMC after a power fail.

4. Special files must be created for each VPM protocol-driver minor device, each trace minor device, and each KMC11:

   ```
   /etc/mknod   /dev/kmc?   c   X   ?
   /etc/mknod   /dev/vpm?   c   Y   ?
   /etc/mknod   /dev/trace  c   Z   0
   ```

   where *X*, *Y*, and *Z* are the major device numbers of the KMC11 driver, VPM protocol driver, and trace driver, respectively. (It is not necessary to make special files for the CSI minor devices since these are never referenced explicitly).

5. Modify the */etc/brc* file so that it executes a **vpmset**(1M) command for each VPM protocol-driver minor device. The arguments to the **vpmset** command specify the protocol-driver minor device, the KMC11, and, if the device is a KMS11, the particular line on the KMS11:

   ```
   /etc/vpmset /dev/vpm? /dev/kmc? [n]
   ```

   KMC11 lines are numbered starting with zero. The **vpmset** commands should be executed on the first transition to multiuser. An attempt to open a VPM protocol-driver minor device for reading and/or writing will fail if the corresponding **vpmset** command has not yet been executed.

Hardware Installation and Switch Settings

The KMC11-B microprocessor and DMC11-DA, -FA, or -MD line unit must be installed in adjacent slots of a PDP-11 or VAX backplane. Care should be taken not to exceed the dc power capacity of the cabinet in which the items are installed. The microprocessor and line unit are interconnected by a 1-foot ribbon cable. The DMC11-DA or -FA line unit is connected to a suitable synchronous modem by a DEC-supplied modem cable. If the HDLC interpreter is used, the modem must be optioned for full-duplex (4-wire) operation; at speeds above 1200 bits per second, this will normally require a private line. The DMC11-DA has an RS-232 interface that is suitable for connection to data sets such as the 208 and 209. The DMC11-FA has a CCITT V.35 interface. The DMC11-MD has an integral 56 kB modem; this unit must be connected by a pair of coaxial cables to another DMC11-MD. The device address and interrupt vector address switches on the KMC11 should be set for the selected addresses. The KMC11 should also be wired for the selected bus priority (normally 5). All switches and jumpers on the DMC11 line unit should be in the normal configuration prescribed by the relevant DEC maintenance manual, but with one exception - the NO CRC switch (switch S2 in switch pack number 1) should be in the ON position. The purpose of this switch setting is to inhibit hardware CRC generation. Hardware CRC generation is not used with the VPM software for this device.

The procedures for installing a KMS11 are similar except that:

1. There are no switches to set on the DMS11-DA 8-line multiplexer.
2. Modem cables are not supplied with the DMS11-DA. Instead, there is an 8-line distribution panel with an RS-232 connector for each line on the DMS11-DA. Each in-use line must be connected by a suitable cable to a synchronous modem. The modem must be optioned for full-duplex (4-wire) operation. The modem should also be optioned to hold the transmitter carrier continuously on because the request-to-send lead to the modem is not controlled by the VPM software for the KMS11. The DM11-BA modem control multiplexer normally included as part of the KMS11 hardware package is not currently used by the VPM software for the KMS11. Limited access to the DM11-BA modem control unit is provided by a separate driver [see **dmk**(7)]. If the modem cannot be optioned for continuous carrier, the *dmk* driver can be used to turn on the request-to-send lead to the modem. For dial-up lines (e.g., a 212 data set operating in synchronous mode), it may be necessary to use the *dmk* driver to turn on the data-terminal ready lead to the modem.

The KMC11 contains a hardware timer which is used by both the VPM and DZ11/KMC11 software. For proper operation with this software, the value of the hardware timer should be 75 microseconds. Newer versions of the KMC11 microprocessor (etch revisions E and later) contain a switch pack at location E82 which is used to select the value of the program timer. E82-8 ON provides a time-out value of 115 *milliseconds;* E82-8 OFF yields the correct value of 75 microseconds. (E82 1 through 7 are not used). On earlier versions of the KMC11, the time-out value is selected by installing the proper value of capacitor C40. For proper operation with VPM and DZ11/KMC11 software, the value of this capacitor should be 4700 pF.

Parallel Communication Link (PCL) Devices

A *PCL* provides the interface for up to two Digital Equipment Corporation PCL11−B network buses. Each bus can be used to interconnect up to 16 CPU's, providing relatively fast communication without individual point-to-point connections.

The interface permits simultaneous bi-directional communication between any machines on a single bus. Additionally, each such path is further subdivided into 8 independent channels. A single control interface is provided to reduce the line monitoring overhead for a daemon process.

Installation

1. Generate a system. For each PCL11−B device, the system description file must contain a line of the form:

   ```
   pcl11b    X    Y    5    Z
   ```

   where X is the transmitter interrupt vector, Y is the UNIBUS address of the transmitter command register, and Z is the number of CPU's attached to that PCL11−B network bus.

   In addition, the PCL *control interface* must be configured into the system. This is done by adding the following line to the system description file:

   ```
   pclctrl    0    0    0    1
   ```

2. Special files must be made for each PCL minor device. Each PCL minor device specifies all information needed to connect to one of the other CPU's attached to the network bus; the minor device number is constructed as follows:

the low order 3 bits	specify a channel number.
the next 4 bits	specify one of 16 machines. (This number must be one less than the PCL11–B receiver's TDM bus address for that machine.)
The next bit	specifies one of the 2 PCL11–B's

The procedure for installing the PCL special files is as follows:

a. mkdir /dev/pcl

b. cd /dev/pcl

c. For each remote CPU attached to the PCL network bus, the following special files are made:

```
/etc/mknod    <machine name>0    c    33    X
/etc/mknod    <machine name>1    c    33    X
/etc/mknod    <machine name>2    c    33    X
/etc/mknod    <machine name>3    c    33    X
/etc/mknod    <machine name>4    c    33    X
/etc/mknod    <machine name>5    c    33    X
/etc/mknod    <machine name>6    c    33    X
/etc/mknod    <machine name>7    c    33    X
```

Where <*machine name*> is the name by which the remote machine is known on the network and X is the minor device as calculated above; the channel number for this calculation is the digit following <*machine name*>.

d. Make the special file for the PCL control interface by executing:

```
/etc/mknod    ctrl    c    34    0
```

e. Finally, give the special files the proper owner, group, and modes.

```
chown    daemon    *
chgrp    daemon    *
chmod    660       *
```

Hardware Installation and Switch Settings

The installation of the PC11–B hardware is documented in Digit al Equipment Corporation publication YC-A20TC-00, "PCL11–B Parallel Communication Link Differential TDM Bus". Because of the design of the UNIX PCL device driver, the transmitter's and receiver's TDM bus addresses must have the same value. The TDM bus addresses are also restricted to the range from 1 to 16.

ADMINISTRATIVE FILES

A. /etc/motd

This file contains the *message-of-the-day*. It is printed by */etc/profile* after every successful *login*; therefore, it should be kept short and to the point.

B. /etc/brc

This file is executed prior to entering any of the numbered *init* states for the first time after a reboot. The file is generally used to clear the file */etc/mnttab* and load KMC11s with their respective scripts. It is important to remember this file is executed once per reboot and is controlled by */etc/inittab*.

C. /etc/powerfail

This shell script is executed according to its line in */etc/inittab*. It is used primarily to reload

KMC11 scripts after a power failure has been detected.

D. /etc/rc

On the transition between init states, */etc/init* executes the shell script */etc/rc* (which must have executable modes). The execution of this file is controlled by a line in */etc/inittab*. For */etc/rc* to properly handle the mounting and unmounting of file systems, the opening of tty lines, etc., it may need certain information that is present in */etc/utmp,* namely, the new (current) state, the number of times this state has previously been entered, and the previous state. The following shell script fragment assigns this data to shell variables and checks for entering init state '2' for the first time. As an example:

```
set `who -r`
cur_mode=$7
no_times=$8
pre_mode=$9
if [  ${cur_mode} = 2 ) -a  ${no_times} = 0 ) ]
then
# commands to be executed when entering multiuser mode
fi
```

Note that these values are carried over between reboots, that is to say, when the system is rebooted into a single user initial state, these values, stored in */etc/utmp* will reflect the state the system was in when it last went down.

Note that the files */etc/rc*, */etc/brc*, */etc/inittab*, */etc/powerfail*, and */etc/shutdown* must be edited to suit local conditions; see **brc**(1M).

E. /etc/inittab

This file is used by */etc/init* to determine which processes to create or terminate in each init state. By convention, state 's' is single user and state '2' is multiuser.

The following line may be used to indicate the default init state, that is, the state the system is to come up in (most likely single user):

```
is:s:initdefault:
```

The following lines arrange for appropriate execution of */etc/brc /etc/rc*, and */etc/powerfail*:

```
bc::bootwait:/etc/brc 1> /dev/console 2>&1
rc::wait:/etc/rc 1> /dev/console 2>&1
pf::powerfail:/etc/powerfail 1> /dev/console 2>&1
```

For line */dev/tty00* for use by 1200 baud asynchronous terminals, add the following:

```
00:2:respawn:/etc/getty -t60 tty00 1200
```

The arguments to getty are the number of seconds to allow before hanging up the line, the device name, optional speed settings which refers to an entry in */etc/gettydefs*, optional type of terminal referenced in **getty**(1M), and optional line discipline.

To add or delete *getty-login* processes while the system is in multiuser mode, make the appropriate changes to */etc/inittab* then issue the command **/etc/init q**. This forces */etc/init* to reread */etc/inittab* without having to change init states.

Again, this file must be edited for local conditions; see **getty**(8), **stgetty**(8), **init**(8), **gettydefs**(4), and **inittab**(5).

F. /etc/passwd

This file is used to describe each *user* to the system. A new line must be added for each new user. Each line has seven fields separated by colons:

1. Login name: normally 1 through 6 characters, first character alphabetic, the remainder alphanumeric, no uppercase characters.
2. Encrypted password: initially null, filled in by **passwd**(1). The encrypted password contains 13 bytes while the actual password is limited to a maximum of 8 bytes. The encrypted password may be followed by a comma and up to four more bytes of password "age" information.
3. User ID: a number between 0 and 65,535; 0 indicates the superuser. User IDs 0 through 99 are reserved.
4. Group ID: the default is group 1 (one). Group IDs 0 through 99 are reserved.
5. Accounting information: this field is used by various accounting programs. It usually contains the user name, department number, and account number.
6. Login directory: full pathname (keep them reasonably short).
7. Program name: if null, */bin/sh* is invoked after a successful login. If present, the named program is invoked in place of */bin/sh*.

For example:

```
ghh::138:1:6824-G.H.Hurtz(4357):/usr/ghh:
grk::244:1:6510-S.P.LeName(4466):/usr/grk:/bin/rsh
```

See also **passwd**(4), **login**(1), and **passwd**(1).

G. /etc/group

This file is used to describe each *group* to the system. The system administrator must add a new line for each new group. Each line has four fields separated by colons:

1. Group name: normally 1 through 6 characters, first character alphabetic, rest alphanumeric, no uppercase characters.
2. Encrypted password: contains 13 bytes while the actual password is limited to a maximum of 8 bytes.
3. Group ID: a number between 0 and 65,535. Group IDs 0 through 99 are reserved.
4. Login names: list of all *login* names in the group, separated by commas; list of all *login* names that may use **newgrp**(1) to become a member of the group.

Group passwords are strongly discouraged. See also **group**(4).

H. /etc/profile

When the shell is executed and is the leader of a process group, as is the case when it is invoked by *login*, it will read and execute the commands in */etc/profile* before executing commands in the user's *.profile* file. This allows the system administrator to set up a standard environment for all users (e.g., executing **umask**, setting shell variables, etc.) and take care of other housekeeping details (such as **news −n**). Note that in */etc/profile* the shell variable $0 indicates the invocation - normal shell (−sh), restricted shell (−rsh), or **su** command (−su).

I. /etc/checklist

This file contains a list of default devices to be checked for consistency by the **fsck**(1M) program. The devices normally correspond to those mounted when the system is in *multiuser* mode. For example, a sample checklist would be:

```
/dev/rp0
/dev/rrp1
```

Note that the *root* device is specified as a *block* device while all others are specified as *character* devices. Character devices can be checked faster than block devices. The *root* device is specified as a block device in order for the **fsck** program to detect when the *root* is being checked, so that any modifications to this file system will result in an immediate reboot request.

J. /etc/shutdown

This file contains procedures to gracefully shut down the system in preparation for file save or scheduled downtime. Beware that no procedures appear after the transition to single user mode as it may not be completed before the transition occurs.

K. /etc/filesave and /etc/tapesave

These files contain prototypes for local file saves.

L. /usr/adm/pacct

This file contains the process accounting information; see **acct**(1M).

M. /usr/lib/acct/holidays

This file defines the local company holiday schedule and designates the start of prime and non-prime processing hours for the accounting system.

N. /etc/wtmp

This file is the log of *login* processes.

REGENERATING SYSTEM SOFTWARE

System source is issued under the directory */usr/src*. The subdirectories are named *cmd* (commands), *lib* (libraries), *uts* (the operating system), *head* (header files), and *stand* (stand-alone programs); see **mk**(8) for details on how to remake system software.

NOTES